A New Approach to the Container Positioning Problem

Ahmt, Jonas; Sigtenbjerggaard, Jonas Skott; Lusby, Richard Martin ; Larsen, Jesper; Ryan, David

Publication date:
2015

Document Version
Peer reviewed version

A New Approach to the Container Positioning Problem*

Jonas Ahmt**, Jonas Skott Sigtenbjerggaard***, Richard Martin Lusby†, Jesper Larsen† and David Ryan‡

**Network Planning, Maersk Line, Copenhagen, Denmark
***Advanced Technology & Architecture, Accenture, Copenhagen, Denmark
†Department of Management Engineering, Technical University of Denmark
‡Department of Engineering Science, University of Auckland

July 6, 2015

Abstract

In this paper the Container Positioning Problem is revisited. This problem arises at busy container terminals and requires one to minimize the use of block cranes in handling the containers that must wait at the terminal until their next means of transportation. We propose a new Mixed Integer Programming model that not only improves on earlier attempts at this problem, but also better reflects reality. In particular, the proposed model adopts a preference to reshuffle containers in line with a just-in-time concept, as it is assumed that data is more accurate the closer to a container's scheduled departure the time is. Other important improvements include a reduction in the model size, and the ability of the model to consider containers initially at the terminal. In addition, we describe several classes of valid inequalities for this new formulation and present a rolling horizon based heuristic for solving larger instances of the problem. We show that this new formulation drastically outperforms previous attempts at the problem through a direct comparison on instances available in the literature. Furthermore, we also show that the rolling horizon based heuristic can further reduce the solution time on the larger of these instances as well as find acceptable solutions to much bigger, artificially generated, instances.

1 Introduction

The shipping industry would not be what it is today without the introduction of the standardized container. This simple box is the focal point of a highly automated transportation system, designed to move freight between any two locations with minimum cost, complication and damage. The introduction of the container has not only made shipping cheap, it has also changed the shape of the world economy (Levinson, 2006).

Over the last couple of decades the demand for container shipments has steadily increased. Furthermore, technological advances have resulted in container vessels that are capable of transporting many more containers than before (see e.g. (Dekker et al., 2006)). This ongoing growth in container shipping has increased the throughput of containers at terminals and put enormous pressure on both ports and terminal operators to increase productivity in order to

*This research has been partially supported by the European Union Seventh Framework Programme (FP7-PEOPLE-2009-IRSES) under grant agreement number 246647 and by the New Zealand Government as part of the OptALI project

be able to handle the significant number of containers in a fast yet efficient way. As a result, terminal operators are now turning to Operations Research (OR) based tools in an attempt to make terminals more efficient (Steenken et al., 2004) and remain competitive.

Container terminals can be typically described as*"open systems of material flow with two external interfaces"* (see (Steenken et al., 2004)). A terminal is therefore a facility where cargo containers are transshipped between different transport vehicles for onward transportation. Although terminals vary all over the world, they all follow a similar concept. Figure 1 illustrates the main characteristics of a terminal.

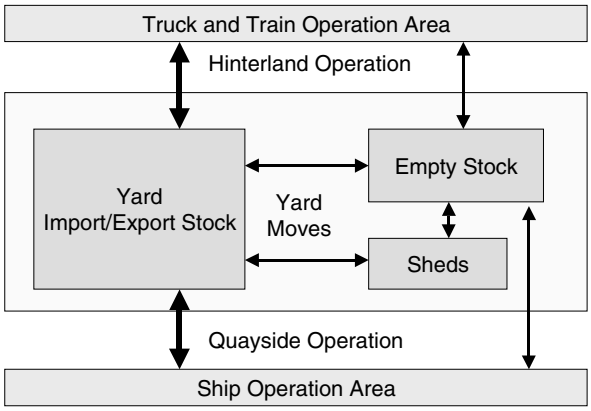

Figure 1: Overview of operations in a container terminal

At container terminals there are several aspects that can be considered and optimized. We focus on one important problem, known as the *Container Positioning Problem* (CPP). After a container has arrived at a terminal, it usually must wait for a period of time before continuing, perhaps by another mode of transportation, on its next journey. How should all containers be handled (i.e. stacked and moved) whilst at the terminal is the solution of the CPP. Ideally, the handling of the containers should be as "efficient" as possible. The term "efficient" is understood to mean in a way that minimizes the movements of the cranes while simultaneously adhering to a number of physical constraints. For instance, the number of containers stacked at any given position cannot exceed a certain number, and containers stacked on top of one another can only be accessed in a Last-In-First-Out (LIFO) order.

This work enhances previous attempts at solving the CPP. We build on the work of (Sibbesen, 2008) and (Phillips, 2009), and, in particular, propose a reformulation of the time discretized version of the CPP (known as the CPPT) presented in (Phillips, 2009). We refer to this new formulation as the CPPTz model since it explicit models the height dimension (or z-coordinate) of the problem. Other modifications to the formulation include modelling "irregularities" such as restricting the set of feasible crane movements and ensuring that a container can never return to its initial position. In addition, we bring the model to a state which more closely reflects reality. The proposed approach limits the number of *reshuffles*, i.e. unproductive moves of a container, by ensuring that early reshuffles are penalized. We also show that our formulation can easily handle an initial state (i.e. unlike previous attempts we do not assume an empty storage yard). The proposed Mixed Integer Programming (MIP) model is solved via a branch-and-cut approach and numerical results show a significant improvement in solution time, while delivering closer to real-life solutions. In addition, we describe sets of valid inequalities, which can be used to strengthen this formulation and present a rolling time horizon based heuristic that makes it possible to solve larger instances of the problem.

The rest of the paper is structured as follows. Section 2 gives a more detailed description of the CPP. This is followed in Section 3 by a review of the existing literature, with a special

focus on the most related work. In Section 4 we present the proposed reformulation of the CPPT model. This new model is not only computationally superior but it also more closely resembles real-life container handling than the traditional CPPT model. Section 5 presents the experimental results of the proposed algorithm on new data sets. This data better reflects reality than the previously available medium-scale test cases. Finally, Section 6 highlights the main conclusions.

2 The Container Positioning Problem

Container storage in a yard is usually separated into different *blocks*, where each block is defined by a number of bays, rows, and tiers. Collectively these, in a sense, provide a 3D coordinate system which can be used to identify particular locations for individual containers. Hence, a block has space for (# of bays)×(# of rows)×(# of tiers) containers. In the literature, the terms *block* and *stack* have been used interchangeably; however, we reserve *stack* to describe a collection of containers placed at a particular bay and row.

Often blocks are separated into different areas for export, import, special and empty containers. Within a block different restrictions can exist based on the type of the container (dangerous goods, reefer etc.). In large container terminals in Europe, the average yard utilization per day is about 15,000-20,000 containers. The storage time of containers in the yard is in the range of 3-5 days on average (Steenken et al., 2004). Blocks may have one or more loading points, separated into arrivals/departures from landside and quayside. Furthermore, each block has one or more associated cranes. There are three different crane types, namely *Rail Mounted Gantry* (RMG) cranes, *Rubber Tyred Gantries* (RTGs) and *Overhead Bridge Cranes* (OBCs). RTGs are more flexible in operation while RMGs are more stable and OBCs are mounted on concrete or steel pillars. The most commonly used gantry cranes span 8-10 rows and allow for a maximum stacking height of between 4 to 10 containers. A block can easily have 37 bays (as is the case in Altenwerder Container Terminal [7]). To increase productivity and reliability, two gantry cranes are often employed at one block. This sometimes calls for a transition area in the block if a container needs to be moved from one side of the block to the other. The performance of gantry cranes depends directly on the size of the block, type of crane and distances to move each container, but a rough estimate on the average is approximately 20 containers per hour (Steenken et al., 2004).

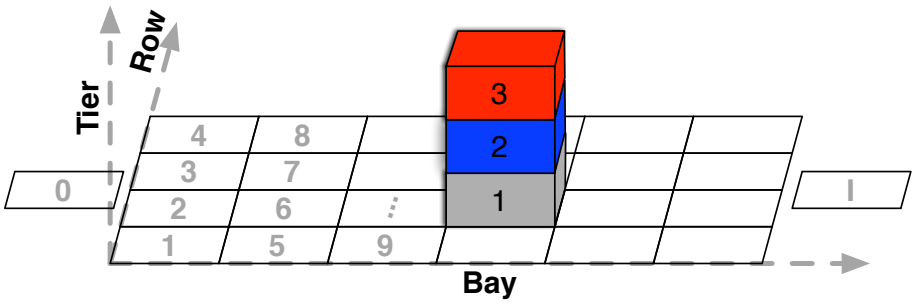

Figure 2: Block layout for the CPP

The CPP deals with the problem of determining the movement of gantry cranes at port terminal yards. A solution to the CPP contains a complete plan of how the containers move through the port terminal, starting from the arrival position at quayside and ending up departing with a vehicle or vessel at a specified departure time at the other end of the block (or vice-versa). Figure 2 provides a schematic overview of the main characteristics of a block.

The CPP contains a number of stack positions; usually the number of which is given by (# of bays)×(# of rows)+2. The additional two positions are the arrival and departure positions respectively. The positions within the block are numbered bay-wise, sequentially from 1 to n; positions 0 and I refer to the arrival and departure positions, respectively (as shown in Figure 2). A container can be placed in different tiers on any given stack position. These are distinguished by numbering the height of the container, starting at one for a container placed directly on the ground.

A stacking decision tool must decide which slot a container is to be assigned to within a block. When a container needs to leave the block for further transportation, there is a risk that it cannot be reached by the crane. Any containers stacked on top of the departing one must be removed first. Even though one may have an optimal stacking plan, data on containers to be stacked could be wrong (or incomplete), or vessel and discharge ports may be changed by the shipping line; this makes the predetermined stacking plan obsolete. About 30-40% of the export containers arriving at European terminals arrive lacking accurate data such as the discharge port or container weight; for import containers, in only about 15% of all cases is the land side mode of transportation known when the container is unloaded from a vessel (Steenken et al., 2004).

The movements of the containers have to be planned with some physical constraints in mind. Perhaps the most important of these is the adherence to the LIFO ordering restriction imposed by container stacks. A container can only be picked up by a gantry crane if it is on top of a stack. This basic physical constraint is what complicates the problem the most; one has to be very careful about the order in which containers are stacked in order to avoid burying containers which are about to depart. Since the arrival and departure dates may change, it is unrealistic to plan placements of containers from the very beginning which will ensure that no reshuffles will be required later. It is, however, desirable to keep the number of these reshuffles to a minimum since it occupies the crane. There is a natural limit to how many containers can be stacked on top of each other. This is governed by the crane type. In this paper we focus on problems containing only one crane, leaving the customization for more specific problems (with e.g. more cranes) as possible extensions of the basic solution method developed.

When a container is moved, it is assumed that it travels the Manhattan distance between the start and end position of the move. In the CPP, the Manhattan distance between two positions is the sum of the number of rows moved and the number of bays moved. The movement time for a container in the CPP stated by Sibbesen (Sibbesen, 2008) is proportional to the Manhattan distance, thereby implying an assumption that the cranes do not move bay- and row-wise at the same time and that the vertical placement (height) is irrelevant for the transportation time. It is also assumed that moving along a row takes the same time as moving up/down a bay, even though the lengths of containers can vary. Finally, each container has an arrival and departure time, which may be not given with initial accuracy.

The objective of the problem is to minimize the amount of time spent on physically moving containers and the number of different positions a container visits during its time in the block. The two parts of the objective are referred to as the transportation time (or movement time) and the number of reshuffles, respectively.

3 Literature Review

In general, literature on descriptive methods, and in particular simulation, dominates this field. As this paper focuses on normative methods for optimizing block operations, the review considers both exact and heuristic approaches.

There are only a few articles before the 90s on normative methods, while numerous papers describe simulation studies and solutions based on control systems. Some of the most active

areas within the container industry are the berth allocation problem (see e.g. (Bierwirth and Meisel, 2010)), stowage plans for container vessels (see e.g. (Delgado et al., 2012)) and the container stacking problem (see e.g. (Dekker et al., 2006)). For a general overview on OR in container terminals, the reader is referred to the excellent surveys (Steenken et al., 2004; Stahlbock and Voß, 2008).

One of the first papers to investigate yard operations in a container terminal is (Cao and Uebe, 1993). The problem of repositioning containers in a storage block is considered, and the authors formulate this as a multi-commodity p-median transportation problem and describe a branch-and-bound approach using a Lagrangian Relaxation. Kim and Kim, in (Kim and Kim, 1999), discuss the problem of minimizing the movement of straddle carriers when used to place containers in blocks. The problem is solved using a Beam Search heuristic. In (Taleb-Ibrahimi et al., 1993) the authors attempt to find the minimal storage space needed to implement a proposed handling and storage strategy given a certain traffic level.

The first use of the term Container Positioning Problem can be found in (Paolucci et al., 1998). In addition to defining the problem, the paper also presents a Mixed Integer Programming (MIP) model. The authors describe the requirements for a management system in connection with the planning of inbound containers. The objective minimizes the total cost of later retrieving the containers again for further transport. In (Bish, 2003; Bish et al., 2005), the CPP is extended to include vehicle scheduling. The solution approach is based on a heuristic and attempts to minimize the time required to unload all containers from a ship berthed in the terminal. In simple settings, most of the developed algorithms find the optimal solutions, while in more general settings, the algorithms obtain near-optimal results for the dispatching problem. In (Bish et al., 2001) the problem is further extended by incorporating quay crane scheduling. The problem is formulated as a transshipment problem and heuristically solved.

In (Preston and Kozan, 2001), a genetic algorithm based approach is proposed for the CPP. The authors consider how to identify the optimal storage strategy with the objective of minimizing the turn around time for ships in the port. In (Kim and Park, 2003) the problem of allocating space to outbound containers is presented and solved using two different heuristics, which are based on the duration of the stay of containers and sub-gradient optimization. Outbound containers are also the focus in the paper (Kim and Lee, 2006); however, the objective is instead to maximize equipment efficiency. In (Kim et al., 2000), the authors describe the problem of minimizing the expected number of container reshuffles when locating outbound containers. Results are based on a dynamic programming algorithm. In (Kang et al., 2006a,b) outbound containers are stacked in order to be efficiently loaded onto a specific single ship. Both papers consider Simulated Annealing approaches and conclude that the described approaches outperform current manual/semi-automatic approaches. In (Park et al., 2013) container reshuffling in a terminal to improve the efficiency of subsequent loading onto a vessel is investigated. Often the time allowed for such preparatory work is limited. A metaheuristic based on cooperative co-evolutionary algorithms is used to identify promising containers to be reshuffled.

The (p)re-marshalling problem entails reshuffling containers in a port yard taking into consideration the departure times of export containers which must be moved to the ships. The objective is to minimize the number of movements. Compared to the CPP, pre-marshalling can be seen as an integrated part, often just looking at a current snapshot and not considering the problem over a wider time horizon. The literature in this field is often based on heuristics, using a multi-commodity time-space network flow model as a basis. An integer programming approach for the problem is described in (Lee and Hsu, 2007). As only a few instances can be solved with the model, a simple two-phase heuristic is devised for real-life instances. In (Exposito-Izquierdo et al., 2012) some simple but efficient algorithms and an instance generator are presented, whereas (Forster and Bortfeldt, 2012) presents a heuristic tree search

procedure. A neighborhood search heuristic for the same problem is presented in (Lee and Chao, 2009). In (Yu and Qi, 2013), overnight re-marshalling is used to prepare storage stacks for subsequent ship operations. This is part of a larger set up that also looks at inbound container moves.

Chen et al. (Chen et al., 2006) formulate a scheduling problem which considers the whole terminal yard. The problem deals with assigning jobs to each piece of equipment in the yard and is solved as a flow shop problem with additional constraints using a Tabu Search heuristic. A more general view on container stacking, i.e. not just outbound containers, is provided by (Kozan and Preston, 2006). In (Kozan and Preston, 2006), two models for transferring and locating containers in the terminal yard are presented. The authors also combine the two models in an iterative search approach and utilize Tabu Search and Simulated Annealing approaches to produce combined solutions. A MIP model, introduced in (Lee et al., 2006), works on an aggregated level and includes the transportation of containers to and from the yard, the storage of unloaded containers and reshuffling. The model determines the minimum number of yard cranes necessary. Due to the complexity of the model, heuristics are developed to solve the test instances. In (Kim and Hong, 2006) the problem of determining locations for relocated containers is investigated. An exact method based on branch-and-bound as well as a heuristic decision rule are devised.

Inter-yard container handling is the focus of (Cordeau et al., 2007). The authors present the problem as a tactical service allocation problem and formulate it as a generalized quadratic assignment problem with side constraints. Two MIP models are also presented. A MIP model for the scheduling of different types of equipment and planning the storage strategy in an integrated way is described in (Wu et al., 2013). As in many other papers, the conclusion is that the MIP models cannot be solved for large scale problems, and a genetic algorithm is developed. The paper does not present evidence of the efficiency of the genetic approach compared to the MIP.

In her doctoral thesis from 2008 (Sibbesen, 2008), Sibbesen describes three different mathematical models used to represent the Container Positioning Problem. The first model, denoted the Container Terminal Problem (CTP), represents the entire port terminal, i.e from the loading and unloading process at quayside, to the storage blocks at the port terminal, and even the loading of the trucks and trains located landside. The author also introduces a second MIP model, the Container Positioning Problem (CPP), which is more suitable for modeling the problem at hand. Some complications in (Sibbesen, 2008; Phillips, 2009) prompted the third model by Sibbesen (Sibbesen, 2008). It is a time-discretized model for the CPP, denoted CPPT, where the time horizon for the problem is split into discretized time steps. Unlike the CPP model, the CPPT model *does* take the stack height limit into consideration, while also making it possible to move several containers at once if more than one crane is present at the port terminal. However, no restrictions are imposed on how multiple cranes can function together.

Phillips (Phillips, 2009) also considers the CPPT model. The existing CPPT model is improved through the removal of redundant constraints, the addition of constraints which were not explicitly stated by Sibbesen, and the removal of the constraint which prevented containers returning to a position previously visited. The basic idea is that most of the LIFO constraints, which make up a large proportion of the total number of constraints in the model, are actually non-binding in an optimal solution. Phillips (Phillips, 2009) manages to identify the important (i.e. binding) LIFO constraints and demonstrates that only a small fraction of these constraints are needed in order to achieve the optimal solution. Most of these essential constraints can be found a priori through analysis of the solution structure. Afterwards, any other LIFO constraint which is violated by the solution to the relaxed problem is added iteratively to the problem until a feasible (and therefore optimal) solution to the original

problem is found. This technique drastically improves the solution speed.

4 A New Formulation for the CPP

In this section we propose a new formulation of the CPP. In all previous models to date, it is assumed that all data is available a priori. In reality, however, this is a dynamic problem, with more information becoming available as time goes by. Therefore, it is meaningless to plan every necessary reshuffle far in advance as other containers may arrive before the departure of the containers for which a meticulous plan has already been made, and thereby disrupt the whole plan. Focus should instead be put on making sure that containers about to leave are able to do so by being on top of a stack when their departure time is due.

The previous section briefly reviewed existing models for the CPP with a special focus on the CPPT models proposed in (Sibbesen, 2008; Phillips, 2009). We specifically highlighted some problems with the model in terms of its computational efficiency and raised doubts as to how accurately it reflects reality. A reformulation of the CPPT model is necessary in order to remedy its bad scalability and slow solution times as well as make it able to solve problems encountered in reality.

4.1 The Mathematical Model

In order to state our proposed formulation of the CPP, we begin by introducing some notation. We assume that the problem is to be solved over a given time horizon, which is discretized into a set, $\mathcal{T} = \{0, \dots, T\}$, of consecutive, non-overlapping *time slots*. We further assume that we have a set $\mathcal{C} = \{1, \dots, C\}$ of containers that require storage in the block during the planning horizon. Each container $c \in \mathcal{C}$ is assumed to have a known arrival time $a_c \in \mathcal{T}$ and a known departure time $d_c \in \mathcal{T}$. The time window $\mathcal{T}_c = \{a_c, \dots, d_c\}$ hence states the set of intervals for which container $c \in \mathcal{C}$ will be in storage. The set $\mathcal{P} = \{1, \dots, P\}$ denotes the set of all positions within the block, and for positions let 0 denote the arrival position and I the departure position. We let $\mathcal{H} = \{1, \dots, H\}$ be the set of tiers in the block. Hence H states the maximum height of any stack. The transportation time between positions $p \in \mathcal{P} \cup \{0\}$ and $q \in \mathcal{P} \cup \{I\}$ (with $p \neq q$) is assumed to be t_{pq}. Recall that there is no transportation time for moving containers vertically. To ease notation, we let $\mathcal{P}_0 = \mathcal{P} \cup \{0\}$, $\mathcal{P}_I = \mathcal{P} \cup \{I\}$, and $\mathcal{I} = \mathcal{P} \cup \{0, I\}$. In stating the formulation, some elements of the sets need to be excluded. For this we introduce the following notation: the superscript $k \rightarrow$ indicates the k first elements are not considered, the superscript $\leftarrow k$ indicates the k last elements are not considered, and the superscript $k \rightarrow l$ indicates the first k and l last elements are not considered. As an example, $\mathcal{T}^{1\rightarrow} = \mathcal{T} \backslash \{0\} = \{1, \dots, T\}$. Additionally, we introduce the following three types of decision variables. The binary decision variable x_{ct}^{ph} states whether or not container $c \in \mathcal{C}$ is placed in tier $h \in \mathcal{H}$ at position $p \in \mathcal{I}$ at time $t \in \mathcal{T}_c$. We define $z_{ctp} \in \{0, 1\}$ to indicate whether container $c \in \mathcal{C}$ is leaving position $p \in \mathcal{P}_0$ at time $t \in \mathcal{T}_c$, or not. Finally, we let $y_{ct} \in \{0, 1\}$, indicate whether container $c \in \mathcal{C}$ is moving at $t \in \mathcal{T}_c$, or not.

Before formulating all constraints mathematically, we provide a brief overview of the constraints that must be enforced when solving the CPP. Firstly, one must ensure that arrival and departure times for each of the containers are respected. Similarly, containers are not allowed to return to the position 0 after arrival, nor can they proceed to position I before departing. Furthermore, one must ensure that a container is either moving or placed in a single position at any given time. Certain capacity restrictions must also be observed. In particular, the capacity of the crane used for moving the containers can never be violated, nor can there be more than one container at any given tier in any given stack. In terms of the LIFO restrictions, one must also ensure that a container stays at the same tier in a stack until it is moved, that

a container is either on the ground or on top of another container at any given time, and that two different containers cannot appear at the same stack and tier one time step apart. Finally, one must also observe a speed restriction for empty crane movements. The objective of the problem, as stated earlier is to minimize the number of reshuffles and transportation time (preferring just in time moves) in handling all containers. A MIP formulation of the CPP is hence given as follows:

$$\min \quad \sum_{c\in\mathcal{C}}\sum_{t\in\mathcal{T}_c}\sum_{p\in\mathcal{P}}\left(\frac{d_c-t}{d_c-a_c}+1\right)z_{ctp}+\sum_{c\in\mathcal{C}}\sum_{t\in\mathcal{T}_c}y_{ct} \tag{1}$$

$$x^{0,1}_{c,a_c}=1 \qquad \forall c\in\mathcal{C} \tag{2}$$

$$x^{I1}_{c,d_c}=1 \qquad \forall c\in\mathcal{C} \tag{3}$$

$$x^{0h}_{ct}=0 \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c^{1\to1}, h\in\mathcal{H} \tag{4}$$

$$x^{Ih}_{ct}=0 \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c^{1\to1}, h\in\mathcal{H} \tag{5}$$

$$\sum_{p\in\mathcal{P}}\sum_{h\in\mathcal{H}}x^{ph}_{ct}+y_{ct}=1 \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c \tag{6}$$

$$\sum_{\substack{q\in\mathcal{P}\\ q\neq p}}\sum_{t'=t}^{t+t_{pq}-1}\sum_{h\in\mathcal{H}}\frac{1}{t_{pq}}x^{qh}_{ct'}\leq 1-z_{c,t-1,p} \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c^{1\to}, p\in\mathcal{P}_0 \tag{7}$$

$$\sum_{c\in\mathcal{C}}y_{ct}\leq 1 \qquad \forall t\in\mathcal{T} \tag{8}$$

$$\sum_{c\in\mathcal{C}}x^{ph}_{ct}\leq 1 \qquad \forall t\in\mathcal{T}, p\in\mathcal{P}, h\in\mathcal{H} \tag{9}$$

$$z_{ctp}\geq\sum_{h\in\mathcal{H}}(x^{ph}_{ct}-x^{ph}_{c,t+1}) \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c^{\to1}, p\in\mathcal{P}_0 \tag{10}$$

$$\sum_{c\in\mathcal{C}}x^{ph}_{ct}\leq\sum_{c\in\mathcal{C}}x^{p,h-1}_{ct} \qquad \forall t\in T^{1\to1}, p\in\mathcal{P}, h\in\mathcal{H}^{1\to} \tag{11}$$

$$x^{ph}_{ct}\leq x^{ph}_{c,t+1}+y_{c,t+1} \qquad \forall c\in\mathcal{C}, t\in T_c^{1\to1}, p\in\mathcal{P}, h\in\mathcal{H} \tag{12}$$

$$x^{ph}_{ct}+\sum_{\substack{c'\in\mathcal{C}\\ c'\neq c}}x^{ph}_{c',t+1}\leq 1 \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c^{1\to1}, p\in\mathcal{P}, h\in\mathcal{H} \tag{13}$$

$$\sum_{c\in\mathcal{C}}\sum_{\substack{q\in\mathcal{P}_I\\ q\neq p}}\sum_{t'=t}^{\min\{t+s_{pq},d_c\}-1}z_{ct'q}\leq\left\lceil\frac{1}{2}\max_{i,j\in\mathcal{P}}s_{ij}\right\rceil\left(1+\sum_{h\in\mathcal{H}}x^{ph}_{ct}-x^{ph}_{c,t+1}\right) \qquad \forall t\in\mathcal{T}^{\to1}, p\in\mathcal{I} \tag{14}$$

$$x^{ph}_{ct}\in\{0,1\} \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c, p\in\mathcal{I}, h\in\mathcal{H} \tag{15}$$

$$z_{ctp}\in\{0,1\} \qquad \forall c\in\mathcal{C}, t\in\mathcal{T}_c, p\in\mathcal{P} \tag{16}$$

$$y_{ct}\in\{0,1\} \qquad \forall c\in C, t\in\mathcal{T}_c \tag{17}$$

The objective function is given by (1). The first part sums the reshuffles, while the second term counts the total transportation time. Without the coefficients on the reshuffling component there would be nothing in the model that would control when reshuffling actually occurs, as is the case with the models proposed in (Sibbesen, 2008; Phillips, 2009). The scaling of the first term is important since it is desired to keep the same balance between the two different terms in the objective function. It should never be preferable to perform two late reshuffles

close to a container's departure if the same result could have been achieved by a single reshuffle earlier in the process. The scaling applied makes sure that the contribution from the reshuffle term is always between one and two. This way, two reshuffles will always add up to a contribution of more than two (since it is the sum of two numbers greater than one) to the objective function, while a single reshuffle will always contribute less than two to the objective function, thereby avoiding the situation where several late reshuffles are preferred to a fewer number of early reshuffles.

Constraints (2) and (3) ensure that each container's arrival and departure times are respected. Containers are prevented from returning to the start after arrival and proceeding to the end before the actual departure time by constraints (4) and (5). Constraint set (6) ensures that a container is either moving or placed at a given position at any time, while constraints (7) ensure the relevant travel time. Constraints (8) and constraints (9) guarantee crane capacity is never exceeded and that there is at most one container on any tier in any position of the block. The necessary link between the z_{ctp} variables and the x^{ph}_{ct}-variables is enforced by constraints (10). All LIFO restrictions are handled by constraints (11)-(13). Constraints (11) ensure there is no gap between containers in a stack; constraints (12) ensure that a container remains at the same height in a stack until it is moved, and constraints (13) guarantee that two different containers cannot appear at the same tier in a given position one time slot apart. Finally, constraints (14) restrict the movement speed of an empty crane. The model already reflects the travel time of the crane with a container. It is necessary to make sure that empty movements with the crane are also modeled. The parameter s_{ij} defines the travel time for an empty crane movement between positions i and j, with $i, j \in \mathcal{P}$ and $i \neq j$. This constraint only becomes active when $x^{ph}_{ct} = 0$ and $x^{ph}_{c,t+1} = 1$, i.e. a container has been dropped off in time t and an empty movement needs to be performed. The constraint makes sure that no container is picked up (i.e. $z_{ct'q} = 0$) from a position until the crane has had time to move to that position. The coefficient $\left\lceil \frac{1}{2} \max_{i,j \in \mathcal{P}}(s_{ij}) \right\rceil$ represents the maximum number of containers which can be picked up. It functions as a "big M" assuring that no constraints are imposed on $x^{m}_{ct'q}$ when the constraint is not active. All decision variable domains are given by constraints (15)-(17).

Model (1)-(17) is similar to previous models for the CPPT. It is, however, noticeably different in one respect; we introduce an additional subscript on the x-variables to model the tier dimension of the problem. In previous attempts, this z-coordinate is not explicitly included. As a result, the tier a container is placed at is not directly represented in any variables of the model. Instead, the number of containers in a stack is determined by counting the number of containers assigned to the stack at any given time while the top container was found by looking at the relative arrival times to the position between the different containers. While resulting in fewer variables, we believe the lack of a z-coordinate (or tier dimension) makes the formulation of the LIFO constraint more complicated. Having access to information about what tier a container is placed in makes it much easier to state the LIFO requirements, and this in fact results in significantly fewer LIFO constraints in the model. Extending the x-variables also makes it much easier to handle containers already placed in the block at the start of the planning horizon; a container can be assigned to a specific height on a stack at the beginning of the planning horizon. There is no need to add additional time slots and extra variables/parameters or apply other work-arounds to represent containers already in the block at the start of the planning horizon.

Adding the tier dimension to the x^{ph}_{ct}-variables will, however, naturally result in more variables for the model. The number of x^{ph}_{ct}-variables will increase by a factor H (number of tiers), where H typically ranges from four to 10. While this is a relatively large increase in the number of variables, the benefits it brings must also be considered – a significant decrease in the number of LIFO constraints and an easier handling of the initial containers. We refer

to this new model as the *CPPTz model* (Time-discretized Container Positioning Problem with z-coordinates). The CPPTz model makes use of the same sets, parameters and decision variables as the modified CPPT model. The main changes are the additions of a set containing the tiers, the corresponding h-index on the x_{ct}^{ph}-variable, and some slight modifications to some constraints. The CPPTz formulation exploits the benefits of both the modified CPPT from (Ahmt and Sigtenbjerggaard, 2010) and the CPPT formulation in (Phillips, 2009). The result is a formulation which is better at capturing the CPP as it exists in reality and with fewer constraints. The formulation does, however, have more variables. How this change in the ratio between variables and constraints affects the computational effort required to obtain a proven optimal solution is difficult to say without extensive testing. Typically, the benefit of removing more constraints will counter the drawbacks of adding extra variables (when only looking at the time it takes to solve the LP-relaxed problem).

As stated above, the CPPTz formulation makes it much easier, and more straightforward, to formulate the LIFO constraints; they are now linear in number with respect to each set. These constraints also allow one to easily formulate the initial state of the block. While these are both nice features, the entire formulation still scales badly with added idle time. This makes the problem more time consuming to solve, even though the complexity of the problem, with respect to the necessary number of reshuffles, has not increased. The main cause of this is the time discretization, which is still part of the CPPTz formulation. The model still contains some symmetry but compared to the original CPPT model significant improvements have been made. The risk of symmetry has been reduced via the reformulation as it allows the model to easily handle initial states. Having containers placed in the block initially will give a natural difference between the positions which did not exist before, thereby reducing symmetry. The structure of the CPP itself *is* very symmetric and it is therefore difficult to decrease symmetry, even if one adopts another formulation of the problem.

4.2 Valid Inequalities

In this section we describe three families of valid inequalities which can be used to tighten the formulation. In other words, these constraints, when added to Model (1)-(17), remove feasible solutions to the LP relaxation; however, they do not change the integer polytope.

The first family of valid inequalities considers the case in which the model fractionally splits a container evenly over the tiers in a stack. A constraint of the following type prevents this form happening.

$$\sum_{c \in \mathcal{C}} x_{c,t-1}^{p,h-1} \geq \sum_{c \in \mathcal{C}} x_{ct}^{ph} \quad \forall t \in \mathcal{T}^{1\rightarrow}, p \in \mathcal{P}, h \in \mathcal{H}^{1\rightarrow} \tag{18}$$

This constraint states that (part of) a container can only be placed in a tier if the tier below had (part of) a container placed there in the previous time slot. Since we assume there is only a single crane in a block, we can also write

$$\sum_{c \in \mathcal{C}} x_{c,t+2}^{p,h-1} \geq \sum_{c \in \mathcal{C}} x_{ct}^{ph} \quad \forall t \in \mathcal{T}^{\leftarrow 2}, p \in \mathcal{P}, h \in \mathcal{H}^{1\rightarrow} \tag{19}$$

This new constraint is similar to (18) and states that (part of) a container can only be placed in a tier if the tier below had (part of) a container placed there two time slots later. This is due to the assumption on the moving speed of the crane; when empty or carrying a container, this is one container length/width per time slot.

If we momentarily accept that it is possible to distribute a container over several tiers (even though this is of course not feasible in an integer solution), this redistribution should at least come with some cost; the cost of picking up the container. The following constraint

ensures that z_{ctp} should also count when part of a container is moved around *within* a stack (i.e. changes tier at the same position).

$$z_{ctp} \geq x^{ph}_{ct} - x^{ph}_{c,t+1} \quad \forall c \in \mathcal{C}, t \in \mathcal{T}_c, p \in \mathcal{I}, h \in \mathcal{H} \tag{20}$$

Furthermore, normally a container (or part of one) should not be able to appear at a position without the container having been moving the time period before. This constraint will replace (11). Constraint (21) takes care of this situation by requiring that (part of) a container arriving to a position was moving the time slot before:

$$\sum_{h \in \mathcal{H}} (x^{ph}_{c,t+1} - x^{ph}_{ct}) \leq y_{ct} \quad \forall c \in \mathcal{C}, t \in \mathcal{T}_c^{1 \to 1}, p \in \mathcal{I} \tag{21}$$

All of the constraints defined above make the solution inherently more integer and hence strengthen the formulation. In Section 5 we assess the impact of including all valid inequalities in the model.

4.3 Solution Method

Given the practical relevance of the problem, short running times are desired. Therefore, where possible, we attempt to reduce the formulation size through the pre-processing of variables. However, for large instances, this is still not sufficient in producing a model that can be solved in reality. Consequently, we propose to solve the full problem using a dynamic rolling horizon approach and argue that this approach is closer to what would be done in reality. We discuss our pre-processing techniques and provide an overview of the rolling horizon approach next.

4.3.1 Pre-processing Variables

Pre-processing can be extremely beneficial in reducing problem size. The aim is to remove redundant variables (and by extension, constraints) whose values can be deduced initially. Such variables would otherwise only contribute to the size of the model. Effective pre-processing can therefore create a smaller problem, which is hopefully faster to solve.

Based on the arrival and departure times of the containers, it is possible to determine the values some variables must take. In particular, and assuming a single crane, one can exclude certain variables from consideration (especially those around the arrival and departure times), i.e. set to the value of zero. Furthermore, we know exactly when certain containers are moving, hence we can set the corresponding y_{ct} values before solving the problem. This is what we refer to as prefixing.

In addition, we can also prefix "non-critical" containers. It can happen that in the considered planning horizon some containers initially placed in the block are not leaving. These are referred to as non-critical containers. If non-critical containers do not conflict with other moves, meaning that they are not placed in a stack above a leaving container, they are not active in the optimization. Nothing is gained by considering them in the optimization, hence they are fixed to their slot for the entire planning horizon. These containers are easy to identify, and fixing them will remove some of the columns in the formulation and speed up the solution process.

4.3.2 Rolling Time Horizon

Previously, the CPP has been treated as a static problem. It was assumed that all information about arriving and departing containers was known in advance, prior to solving the problem. Furthermore, all planning started from an empty block with no stacks of containers at any

position. As mentioned previously, these assumptions are unrealistic and do not accurately reflect how the problem arises in real life. We therefore introduce a dynamic approach using a rolling time horizon. Not only does a rolling horizon make it possible to handle dynamic data sets, but it also allows CPPs to be solved much faster than has previously been possible.

As the name suggests, the approach makes use of a planning horizon that moves forward in time. The planning horizon considered at any one time is typically much shorter than the full horizon and considers only those containers that are of importance during the considered time interval. Figure 3 illustrates this concept. The idea is that the start of the planning horizon is the actual time at the container terminal; this then naturally moves forward as time progresses. The solver is only called anew when new information enters the planning horizon. This can either be the arrival or the departure of a new container. The planning horizon should be long enough to capture all events that are certain, ensuring that no unnecessary work is done to place containers at a favorable position. It should also be long enough to ensure feasibility for the entire planning horizon. Solving overlapping, shorter time horizons, can produce a situation in which the CPP defined for a later time horizon is infeasible, despite the fact that all previously considered time horizons yielded a feasible solution. This is a natural consequence of a sequential approach. Infeasibility typically occurs when there is insufficient time to reshuffle the containers to get to a departing container. To increase the likelihood of finding feasible solutions in later time horizons, we introduce a term in the objective function (1) which focuses on containers that have arrived in the yard (or which have been initially positioned there), but which are due to depart after the end of the current time horizon. Ideally, these containers should be positioned close to the top of a stack and located close to the departure position at the end of the current horizon. Let us suppose we are solving the CPPTz for a given horizon $w = [t_s, t_e]$, where t_s and t_e denote the delimiting time intervals, and that there is a set of departing containers (already present in the yard or arriving at latest at time t_e) $\mathcal{C}_D^w$, where each $c \in \mathcal{C}_D^w$ has a departure time d_c greater than t_e. The following term penalizes containers in $\mathcal{C}_D^w$ that are deep in stacks and far from the departure position and is added to (1).

$$\sum_{c \in \mathcal{C}_D^w} \sum_{p \in \mathcal{P}} \sum_{h \in \mathcal{H}} \rho_{cph} x_{ct_e}^{ph}, \tag{22}$$

where ρ_{cph} is the chosen penalty for placing container c in tier h of position p at time t_e. This penalty should reflect the order of the departing containers. That is, less penalty should be incurred by placing the container that is due to depart next higher in the stack than one which is due to leave later. Also, less penalty should be incurred the closer the container is to the departure position. With this term added to the objective function, the containers should arrange themselves in a manner which allows easier unloading in later horizons.

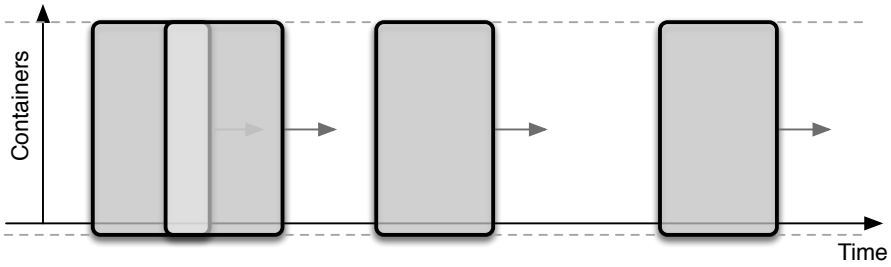

Figure 3: Rolling time horizon methodology

Within the horizon the main focus should be put on the containers which are about to depart

since these are the most essential to focus on at any given point. The placement and reshuffling of all other containers is almost irrelevant as long as it is made sure that none of these are moved in a way which hinders the process of getting the soon-to-be-departing containers to the top of a stack so that the departing containers can easily leave the block in due time.
In what follows, more specific details are given on the different parts of the solution algorithm. The method can be described as a three step process:

1. **Initialization:**
   All data which is available at the beginning is used to create variables and parameters representing initial stacks and arriving and departing containers. An estimated duration for how long it will take to get a container to the top of a stack is needed (length of the rolling time horizon). If any container is leaving within this interval, it is marked as critical, otherwise it is considered non-critical.

2. **Solve current time horizon:**
   The problem is solved using the CPPTz model with the important difference that only the critical containers have the criterion that they should leave during the horizon.

3. **Locate next time for reoptimization:** The next time to call the solver for reoptimizatiton can be caused by three different situations:

   (a) **Information about a new container arrival becomes available:** If the newly arriving container is not leaving within the considered horizon, every variable in the current solution is fixed so that the containers maintain their positions and scheduled movements.

   (b) **A non-critical container's departure time enters the time horizon:** The departing container is marked as critical and the departure criterion for the container is included in the model (again, this will happen automatically since only constraints within the considered time horizon are set up). The time horizon in the next iteration will end with this container departing.

   (c) **A container is moving at the end of the time horizon:** If a solution is found for a given horizon in which it has been planned that a container should be moving at the end of the time horizon, this could mean that the container in question is on its way to be placed at the first position after its arrival.

The algorithm enters an idle state when no more information about arriving or departing containers becomes available. In reality, the solution process would never stop since new containers will always arrive or depart at some point. For more details on the implementation of the rolling time horizon approach the reader is referred to (Ahmt and Sigtenbjerggaard, 2010). Figure 4, does however, schematically illustrate the solution process for one of the problem instances (`5S.tm` with a rolling time horizon of length 16 time slots).

The numbers indicate the terminal position; i.e. numbers 0 and 4 refer to the arrival and departure positions, respectively. Initially the arrival of Container One (C1) at position 0 triggers a calculation for the entire length of the time horizon (which is 16). In this small example all containers actually arrive within this first rolling time horizon, and therefore no further arrivals trigger any optimization. Instead, the remaining executions of the model are due to a departing container (i.e. a container at position 4). These appear in bold face in the figure. The first time this happens is during time slot 22, as shown in the second box in Figure 4. Calculations are performed for the preceding 16 time slots, while the solution for the first six time slots is fixed (shaded dark gray). In each horizon there is a row for each container,

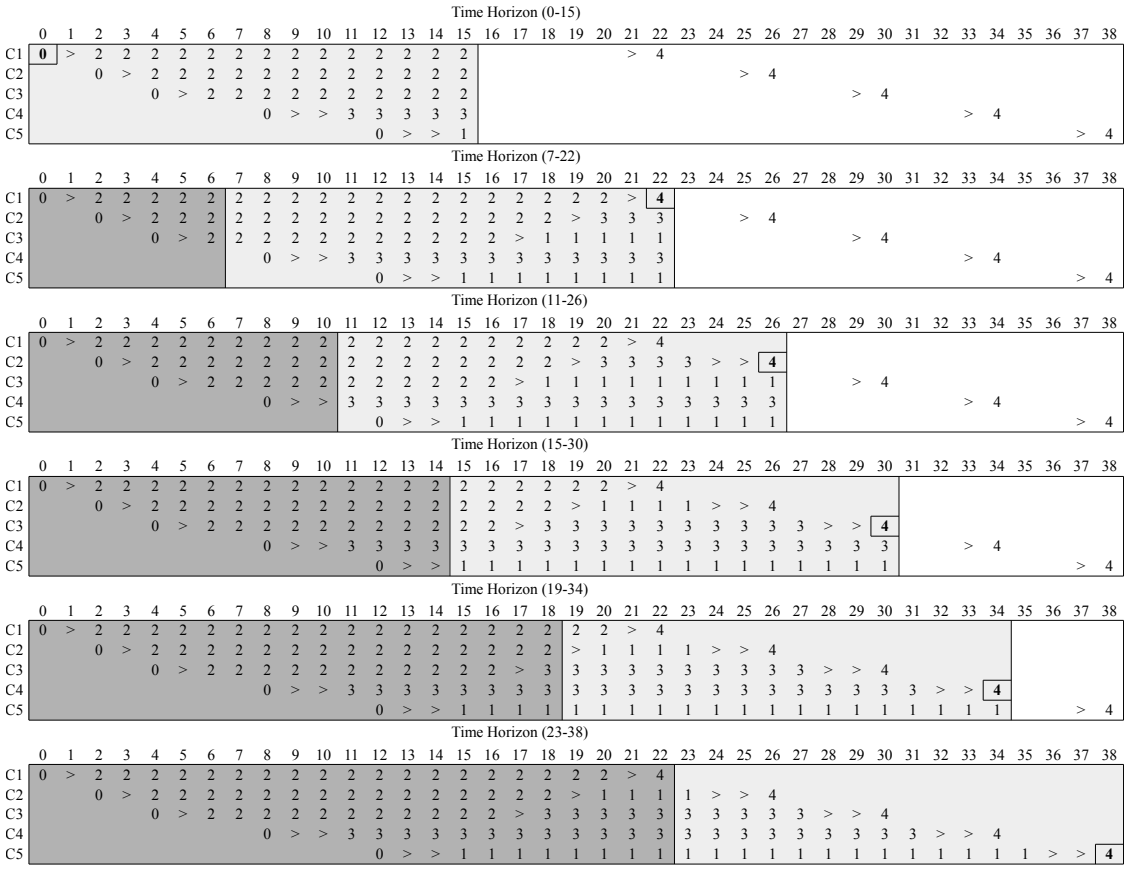

Figure 4: 5S.tm solved using rolling time horizon of length 16

while a ">" indicates that the container is moving. In each of the horizons, the time slots under consideration are shaded light gray. Each time horizon overlaps with the previous one.

From a computational perspective, solving the CPP using the proposed rolling time horizon approach provides both advantages and disadvantages. By far the most important benefit comes from the fact that this solution process is much closer to reality than solving the whole problem at once assuming all information is available and accurate. In addition, splitting the problem into smaller time horizons creates a number of smaller sized problems which are much faster to solve; even if the problem is solved using several overlapping time horizons there is a good chance of ending up with a better total solution time than if the problem was solved just once using the whole time span as the perspective. The main drawback when splitting up the problem into smaller time horizons is that we might suboptimize and therefore damage the quality of the solution.

5 Computational Results

In this section we analyse the performance of both our MIP reformulation of the CPPT and the rolling time horizon based heuristic on several data sets. To provide an accurate comparison with previous attempts at this problem, several of the data sets are those used in the computational study of (Phillips, 2009). This allows direct comparisons to be made. For such instances, the impact of the valid inequalities discussed in Section 4.2 on the lower bound is also reported. The remaining instances, while based on reality, are artificial data sets. These are significantly larger (in terms of the number of containers and the length of the planning horizon) and have been included in an attempt to reflect problems that are encountered in

reality.

5.1 Test Cases

We begin with a short description of the data sets used in (Phillips, 2009). An overview of the instances is given in Table 1. It should be noted that these instances, with the exception of 7S.tm, first appeared in (Sibbesen, 2008). Many other instances were also considered in (Sibbesen, 2008); however, we restrict our attention to only those considered in (Phillips, 2009). Furthermore, the integer programming approaches presented in (Sibbesen, 2008), were already unable to solve instance 6S.tm. As empty crane speed was ignored in both (Phillips, 2009) and (Sibbesen, 2008), some modification to the data sets was needed to ensure the instances were still feasible when enforcing a restriction on empty crane speed. Without such a restriction a crane could move "instantaneously" between any two positions if not moving a container. As such, we simply inserted sufficient idle time into the planning horizon to allow an empty crane to reach pick up points.

The naming of each instance provides some of the characteristics of the respective problem. The first number indicates the number of containers in the problem, the next character specifies the arriving/departing structure ("S" is for staircase). In other words, the containers depart the yard in the order they arrive. The third character indicates whether the problem is tight (t) or scaled (s), referring to the time dimension of the problem. Tight instances typically have the minimum feasible planning horizon, while scaled instances have extra idle time. Finally the optional "m" at the end states whether or not we have modified the instance to ensure empty crane movements are feasible. The new planning horizon (as well as the original horizon) are given. Note that only instance `4S.s` has enough idle time between events and is therefore not changed. The impact on the horizon length by adding the time slots can also be seen.

Problem	Rows	Bays	Tiers	Containers	Horizon	Org. horizon
`4S.tm`	2	1	3	4	28	18
`4S.s`	2	1	3	4	54	54
`5S.tm`	3	1	3	5	38	24
`5S.sm`	3	1	3	5	72	66
`6S.tm`	3	1	3	6	46	28
`7S.tm`	2	2	3	7	58	34

Table 1: Test instances.

One can observe that these instances are quite small. Consequently, we have generated larger, more realistic data sets. The five different instances created are of increasing size and complexity. They have been generated based on (Sibbesen, 2008), where real-life data sets were used to test the heuristic methods proposed by the author. The data was provided by the Oslo Container Terminal (OCT), and consists of containers that departed from a single block in the terminal during November 2005. All generated instances have been created with the aim of producing data instances that reflect reality. A detailed description on exactly how the instances have been generated can be found in (Ahmt and Sigtenbjerggaard, 2010). Table 2 lists the specifications of each created instance.

The instances are named in such a way that the three main characteristics of an instance to the CPP are reflected. For example, instance `3x4x3|50|80` indicates that the problem contains a block of 3 rows, 4 bays, and 3 tiers. The number 50 denotes the block usage, i.e. the percentage of the number of slots in the block that are occupied by containers on average during the planning horizon. Finally, the number 80 refers to the length of the planning

Case	Block info				Container info			Horizon
	Rows	Bays	Height	Usage	Ini.	Arr.	Dep.	
`3x4x3|50|80`	3	4	3	50	17	3	6	80
`3x4x3|65|80`	3	4	3	65	22	6	5	80
`4x6x4|50|80`	4	6	3	50	47	5	3	80
`4x6x4|50|160`	4	6	3	50	47	6	4	160
`4x7x4|50|130`	4	7	3	50	72	3	6	130

Table 2: The five medium scale instances.

horizon. It is worth mentioning that while these data instances are a big improvement over previous instances which operate on block sizes of at most $2 \times 2 \times 3$ (rows $\times$ bays $\times$ tiers), they still do not have a size that is encountered in reality. As an example, the data sets from Oslo report block sizes ranging from $7 \times 14 \times 5$ to $8 \times 27 \times 5$.

5.2 Results

We consider the instances in Table 2 first and assess the performance of the rolling time horizon based heuristic. Note that these instances are too large to be solved efficiently with the MIP formulation. In determining the "best" settings for the rolling time horizon, we toggle various parameters. In particular, we compare rolling horizon lengths (20 or 30 time slots), whether or not to add the sets of valid inequalities from Section 4.2 (T), and whether or not providing the MIP with a feasible solution to begin with is advantageous (i.e. utilize the CPLEX `MIPstart` feature). We refer to the latter as the rolling horizon approach with warm starts (W). A "✓" is recorded in the appropriate column if the parameter is activated. All tests have been performed on a Lenovo x220 ThinkPad running Ubuntu Linux 13.10 with an Intel(R) Core(TM) i7-2640M CPU, 2.80GHz, and having 8GB RAM. Version 12.5 of the commercial solver CPLEX (multiple thread) has been used to solve the integer programs.

Table 3 provides a summary of the results. For each instance and each horizon length the following information is given: the objective value (in terms of moving time and reshuffles) $\mathcal{Z}$, the total moving time $\mathcal{M}$, the number of reshuffles $\mathcal{R}$, the total runtime (in seconds) t, and the average run time of each horizon $\hat{t}$, again in seconds. Note that the objective value of a single MIP can be different, even though the transportation time and the number of moves is identical (see e.g. instance `3x4x3|50|80`). The difference in objective function value merely indicates a time discrepancy between container movement times. As mentioned in Section 4.3, an additional term, (22), was added to the objective in order to reduce the risk of infeasibility for the heuristic. This value is not counted in the reported objective value. From Table 3 it can be seen that there is a noticeable difference in the solution time as the horizon length increases. This is illustrated quite clearly from the first instance (`3x4x3|50|80`) where the solution times have increased by 942%, 1202%, 1102% and 1362%, respectively, for each of the different settings. All other instances and settings also have significant increases in solution times, although not all of them show the same extreme increases as the first instance. It does not come as a surprise that the solution time increases as the rolling horizon gets longer, as it is expected to increase much more than linearly for MIP problems such as the CPPTz. Furthermore, due to the heuristic nature of the approach, a better solution cannot be guaranteed when considering a longer horizon; however, at least the possibility is there. The reduction in solution for a shorter horizon more than compensates for the extra number of horizons that need to be solved, and the results suggest that, perhaps with the exception of (`3x4x3|50|80`), solution quality is not overly compromised.The number of time horizons

			Horizon length = 20					Horizon length = 30				
	T	W	$\mathcal{Z}$	$\mathcal{M}$	$\mathcal{R}$	t	$\hat{t}$	$\mathcal{Z}$	$\mathcal{M}$	$\mathcal{R}$	t	$\hat{t}$
3x4x3|50|80	-	-	40.269	23	13	22.13	3.69	32.848	18	11	208.42	41.68
	✓		35.576	21	11	23.63	3.94	37.890	22	12	284.01	56.80
		✓	35.576	21	11	22.61	3.77	32.848	18	11	249.18	49.84
	✓	✓	35.751	21	11	26.02	4.34	37.890	22	12	354.34	70.87
3x4x3|65|80			38.281	25	10	17.23	3.45	38.281	25	10	69.23	13.95
	✓		38.281	25	10	16.73	3.35	38.281	25	10	86.28	17.26
		✓	38.281	25	10	17.19	3.44	38.281	25	10	78.28	15.66
	✓	✓	38.281	25	10	18.36	3.67	38.281	25	10	90.88	18.18
4x6x4|50|80			19.214	14	4	43.67	8.73	19.214	14	4	226.15	56.54
	✓		19.214	14	4	40.99	8.20	19.214	14	4	268.01	67.00
		✓	19.214	14	4	54.82	10.96	19.214	14	4	230.18	57.55
	✓	✓	19.214	14	4	54.72	10.94	19.214	14	4	229.55	57.39
4x6x4|50|160			25.945	19	6	90.65	15.11	28.307	20	7	325.44	89.12
	✓		28.307	20	7	96.61	16.10	28.307	20	7	550.87	96.99
		✓	28.307	20	7	130.25	21.71	28.307	20	7	735.64	122.61
	✓	✓	25.945	19	6	111.33	18.55	28.307	20	7	654.65	90.69
3x7x4|50|130			42.081	28	11	173.28	28.88	42.081	28	11	1166.74	233.35
	✓		42.081	28	11	165.46	27.58	44.204	29	12	1956.72	391.34
		✓	44.204	29	12	288.93	48.16	42.081	28	11	1633.94	326.79
	✓	✓	42.081	28	11	165.99	35.61	42.081	28	11	1431.03	286.21

Table 3: Results for different parameter settings

solved for a length of 20 time slots compared to a length of 30 time slots does not differ much for these instances; five to six shorter horizons are solved for each case with a rolling horizon of length 20, while four to six are solved using a horizon length of 30. In instances `3x4x3|65|80` and `4x6x4|50|160` there is no difference in the number of time subproblems solved for the two different cases. The importance of choosing as short a time horizon as possible is thus illustrated quite clearly from the results.

Looking at Table 3 again, it would appear that warm starts do not seem to be advantageous in the current implementation. Long feasibility checks, wasted attempts to find solutions and memory build-ups are to blame, and in many cases with a rolling horizon of length 30 the use of the warm start actually seems to slow down the solution process. In these cases, the building of the constraints takes a significant amount of time due to the size of the problems, which means that solving overlapping horizons twice drastically impedes the performance, even if the actual solution process itself is fast for the subproblems with many fixed variables.

Tightening is also not that beneficial from a computational perspective. With the exception of the run with a time horizon of 30 time slots and tightening activated for instance `4x6x4|50|160`, all tests using tightening performed worse than when it was not used. This is most probably a direct result of the fact that, while the tightening constraints do have an integerizing effect, they are typically large in number, thus increasing the size of the MIP models and impede the performance of the solver. As an example, consider Table 4. This states the number of constraints in the first horizon of the first instance with and without tightening for a rolling horizon of length 30. An interesting direction for future research could be therefore to dynamically separate such constraints.

Constraint	Tally	Total (%)	Cum. (%)
(12)	18828	33.34	33.34
(13)	18144	32.13	65.46
(7)	6799	12.04	77.50
(10)	6799	12.04	89.54
(4)	1563	2.77	92.31
(5)	1563	2.77	95.07
(9)	1080	1.91	96.99
(11)	720	1.27	98.26
(6)	542	0.96	99.22
(14)	406	0.72	99.94
(8)	30	0.05	99.99
(2)	2	0.00	100.00
(3)	2	0.00	100.00
Summed	56478		

(a) horizon length = 30, no tightening

Constraint	Tally	Total (%)	Cum. (%)
(20)	20397	24.26	24.26
(12)	18828	22.39	46.65
(13)	18144	21.58	68.24
(7)	6799	8.09	76.32
(10)	6799	8.09	84.41
(21)	6552	7.79	92.20
(4)	1563	1.86	94.06
(5)	1563	1.86	95.92
(9)	1080	1.28	97.20
(18)	696	0.83	98.03
(19)	672	0.80	98.83
(6)	542	0.64	99.48
(14)	406	0.48	99.96
(8)	30	0.04	100.00
(2)	2	0.00	100.00
(3)	2	0.00	100.00
Summed	84075		

(b) horizon length = 30, tightening

Table 4: Constraint info for first horizon of `3x4x3|50|80`.

5.3 Comparison with Modified CPPT

With the testing of the heuristic complete, we turn our attention to the comparison with previous approaches for the CPP, in particular the computational study of (Phillips, 2009). That is, we compare the performance of the reformulation, along with the heuristic, on the instances presented in Table 1. First, we provide a short overview of the impact of the tightening constraints on these problems. The results are given in Table 5. The first column states the problem instance, while columns LP^* and LP_t^* give the optimal LP solution of the CPPTz model with and without the tightening constraints. The last column gives the percentage increase in the lower bound. Improvements of 8-17% suggest such constraints have some impact on the bound and should be be considered for inclusion. For these instances, there is negligible difference in the running times due to the size of the problems. For larger problems it is impractical to add all a priori, and, as has already been mentioned, a dynamic separation routine might be worth considering.

	LP^*	LP_t^*	%
4S.tm	14.472	16.926	16.96
4S.s	14.214	16.179	13.83
5S.tm	19.482	21.512	10.42
5S.sm	19.211	21.292	10.83
6S.tm	24.503	26.677	8.87
7S.tm	30.476	33.067	8.50

Table 5: Impact of valid inequalities on lower bound

	(Phillips, 2009)			Full CPPTz MIP				CPPTz Rolling Horizon			
	$\mathcal{M}$	$\mathcal{R}$	t	$\mathcal{Z}$	$\mathcal{M}$	$\mathcal{R}$	t	$\mathcal{Z}$	$\mathcal{M}$	$\mathcal{R}$	t
4S.tm	14	6	4.66	21.722	14	6	0.50	21.722	14	6	0.46
4S.s	14	5 (*7)	23.14	20.920	14	6	5.24	20.920	14	6	4.61
5S.tm	18	7	29.52	26.434	18	7	1.65	27.588	19	7	6.00
5S.sm	18	7 (*9)	493.10	25.712	18	7	116.55	25.712	18	7	6.26
6S.tm	22	9 (*11)	292.40	32.962	22	9	3.23	32.962	22	9	8.98
7S.tm	23*	11	3337.08	40.366	27	11	311.20	40.366	27	11	18.68

Table 6: Comparison of MIP, rolling horizon, and methodology in (Phillips, 2009)

Table 6 provides a comparison of the full MIP solve with the rolling time horizon based approach. For the rolling horizon, a planning horizon of 25 time slots was used and all tightening constraints added. No warm starts were used. This table also provides the solutions (and run time) obtained using (Phillips, 2009). Results from (Sibbesen, 2008) have not been included as this is consistently worse on all tests made in(Phillips, 2009).We note that the results for the CPPT model in (Phillips, 2009) have been obtained using a PC with an Intel Core 2 CPU, 2 $\times$ 1.86 GHz and 2 GB RAM running Windows 7 Enterprise on 32 bit, with CPLEX 12.0 as the solver running on two cores.

Regarding the length of the planning horizon, a too short planning horizon might lead to infeasiblity as the model cannot recover the containers in due time. For example, if a planning horizon of 20 time slots is used, then then a solution cannot be found for instance 5S.sm. Making the planning horizon longer increases the possibility of finding feasible solutions; however, it also creates longer running times. From a dynamic update perspective, one must therefore make sure enough future information is considered to ensure departing containers can be retrieved in time.

It can be seen that the rolling time horizon approach only decreases the solution time for the larger problems (i.e. 5S.tm and 7S.tm). This is not surprising. In the case of the smaller problems, the overhead in using a rolling horizon outweighs the benefit in solving smaller horizons. For `7S.tm` the significance of the rolling horizon is seen as the solution time decreases from 311 seconds to approximately 19 seconds, while still finding the optimal solution. Note that there is no guarantee the heuristic actually produces the optimal solution (as can be seen for instance 5S.tm). For even bigger problems, such as those in Table 2, the rolling time horizon approach makes it possible to quickly find acceptable solutions that would otherwise take much longer than a full MIP solve. The results suggest that a direct MIP solve on the full horizon using the CPPTz model is superior to the CPPT model reported in (Phillips, 2009). In almost all cases the run time difference is dramatic. Furthermore, the CPPTz actually finds better solutions in three of the cases (those marked with an asterisk in the reshuffle column). This is due to the fact that a container reshuffle was not counted if it returned to the same position in (Phillips, 2009). The actual number of reshuffles is given in parenthesis. For the last instance, the difference stems from the fact that no empty crane speed restriction is enforced in (Phillips, 2009). One of the major advantages of the rolling time horizon based heuristic is that long periods of idle time at the terminal can easily be ignored.

6 Conclusions

In conclusion, this paper proposes a new time-discretized model for the Container Positioning Problem. We have demonstrated that this model, together with the rolling time horizon based solution approach, is to date the most efficient mathematical programming method for solving this problem. Results indicate that it significantly outperforms previous approaches. The modifications we have made in the reformulation have resulted in significant improvements in the solution time of the small test problems that were also used by (Phillips, 2009; Sibbesen, 2008). This is somewhat pleasing, given that this was not the primary purpose. Rather the reformulation was an attempt to produce a model that better reflected reality. We have further demonstrated that the CPPTz model scales much better, especially with respect to the LIFO constraints, which became easier to formulate. The new formulation also made the handling of containers, initially placed in the block at the start of the planning horizon possible in an intuitive way.

Aside from this new formulation, several improvements in the solution method were introduced. Prefixing reduces the number of variables in the problem by anticipating certain values of the decision variables based on the arrival and departure times of the containers. While non-critical container fixing further decreased the number of variables by ensuring certain containers maintained the position they were placed in at the beginning of the planning horizon, if it could be ascertained that they would not need to move. Valid inequalities were also added to the CPPTz model to tighten the formulation. We showed that such valid inequalities improve the lower bound significantly; however, the number of such constraints were typically large, negatively effecting the run time. Dynamically separating such constraints is a promising direction for future research.

Perhaps the most important factor in the solution process was the implementation of the rolling time horizon. By splitting up the problem into multiple subproblems, a solution can be achieved much faster, while still providing acceptable solutions. This approach also better reflects the real application as it is only reasonable to plan for containers for which the information about arrival or departure times is known with certainty.

For larger problems, the a rolling time horizon appears to be extremely useful, perhaps even a necessity. Prefixing containers and non-critical container fixing is also required to

produce acceptable running times; The tests demonstrated the importance of choosing an appropriate horizon length, as the solution speed proved to be very dependent on the length of the horizon. The proposed methodology has made it possible to solve much more realistic problem instances with mathematical programming techniques.

Acknowledgments

The authors would like to thank Antony Phillips (University of Auckland) for providing us with his thesis ((Phillips, 2009)), Python code and all test cases. In addition, the authors would also like to thank Finn Nørgaard and Carsten Gitter from InPort. Finn and Carsten were helpful with providing information to get a better understanding of the real-life challenges in the CPP.

References

Jonas Ahmt and Jonas Skott Sigtenbjerggaard. A new approach to the container positioning problem. Master's thesis, Technical University of Denmark, 2010.

Christian Bierwirth and Frank Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615 – 627, 2010.

E. Bish. A multiple-crane-constrained scheduling problem in a container terminal. *European Journal of Operational Research*, 144:83 – 107, 2003.

Ebru K. Bish, Thin-Yin Leong, Chung-Lun Li, Jonathan W. C. Ng, and David Simchi-Levi. Analysis of a new vehicle scheduling and location problem. *Naval Research Logistics*, 48(5): 363 – 385, 2001.

E.K. Bish, F.Y. Chen, Y.T. Leong, B.L. Nelson, J.W.C. Ng, and D. Simchi-Levi. Dispatching vehicles in a mega container terminal. *OR Spectrum*, 27(4):491–506, 2005.

B. Cao and G. Uebe. An algorithm for solving capacitated multicommodity p-median transportation problems. *Journal of the Operational Research Society*, 44(3):259 – 269, 1993.

Lu Chen, Nathalie Bostel, Pierre Dejax, Jianguo Cai, and Lifeng Xi. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research*, 181:40 – 58, 2006.

J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia. The service allocation problem at Gioia Tauro maritime terminal. *European Journal of Operational Research*, 176:1167 – 1184, 2007.

R. Dekker, P. Voogd, and E. Asperen. Advanced methods for container stacking. *OR Spectrum*, 28:563 – 586, 2006.

Alberto Delgado, Rune Møller Jensen, Kira Janstrup, Trine Høyer Roser, and Kent Høj Andersen. A constraint programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research*, 220(1):251 – 261, 2012.

C. Exposito-Izquierdo, B. Melian-Batista, and M. Moreno-Vega. Pre-marshalling problem: Heuristic solution method and instances generator. *Expert Systems with Applications*, 39 (9):8337–8349, 2012.

F. Forster and A. Bortfeldt. A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39(2):299–309, 2012.

J. Kang, M.-S. Oh, E. Y. Ahn, K. R. Ryu, and K. H. Kim. Planning for intra-block remarshaling in a container terminal. In *Advances in applied artificial intelligence*, volume 4031, pages 1211 – 1220. Springer, 2006a.

J. Kang, K. R. Ryu, and K. H. Kim. Determination of storage locations for incoming containers of uncertain weight. *Advances in applied artificial intelligence*, 4031:1159 – 1168, 2006b.

K. Kim, Y. Park, and K. Ryu. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124:89 – 101, 2000.

K. H. Kim and G.-P. Hong. A heuristic rule for relocating blocks. *Computers & Operations Research*, 33:940 – 954, 2006.

K. H. Kim and J.-S. Lee. Satisfying constraints for locating export containers in port container terminals. *Computational Science and its application*, 3982:564 – 573, 2006.

K. H. Kim and K. T. Park. A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research*, 148:92 – 101, 2003.

Kap Hwan Kim and Ki Young Kim. Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers & Industrial Engineering*, 36:109 – 136, 1999.

E. Kozan and P. Preston. Mathematical modeling of container transfers and storage locations at seaport terminals. *European Journal of Operational Research*, 28:519 – 537, 2006.

L. H. Lee, E. P. Chew, K. C. Tan, and Y. Han. An optimization model for storage yard management in transshipment hubs. *OR Spectrum*, 28:539 – 561, 2006.

Y. Lee and S.-L. Chao. A neighborhood search heuristic for pre-marshalling export containers. *European Journal of Operational Research*, 196(2):468–475, 2009.

Y. Lee and N.-Y. Hsu. An optimization model for the container pre-marshalling problem. *Computers & Operations Research*, 34:3295 – 3313, 2007.

M. Levinson. *The Box: How the Shipping Container Made the World Smaller and The World Economy Bigger*. Princeton University Press, 2006.

M. Paolucci, V. Recagno, and S. Sacone. Designing container yard management systems. In *Proceedings of the International Conference on Computer Aided Design, Manufacturing and Operation in the Railway and Other Advanced Mass Transit Systems*, pages 351 – 360, 1998.

K. Park, T. Park, and K. R. Ryu. Planning for selective remarshaling in an automated container terminal using coevolutionary algorithms. *International Journal of Industrial Engineering*, 20(1-2):176, 2013.

A. Phillips. Optimisation models and methods for the container positioning problem in port terminals. Technical report, University of Auckland, 2009.

P. Preston and E. Kozan. An approach to determine storage location of containers at seaport terminals. *Computers & Operations Research*, 28:983 – 995, 2001.

Louise K. Sibbesen. *Mathematical Models and Heuristic Solutions for Container Positioning Problems in Port Terminals*. PhD thesis, Technical University of Denmark, 2008.

Robert Stahlbock and Stefan Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30:1 – 52, 2008.

D. Steenken, S. Voß, and R. Stahlbock. Container terminal operation and operations research - a classification and literature review. *OR Spectrum*, 26:3 – 49, 2004.

Mounira Taleb-Ibrahimi, Bernardo de Castilho, and Carlos F. Daganzo. Storage space vs handling work in container terminals. *Transportation Research Part B: Methodological*, 27: 13 – 32, 1993.

Y. Wu, J. Luo, D. Zhang, and M. Dong. An integrated programming model for storage management and vehicle scheduling at container terminals. *Research in Transportation Economics*, 42(1):13–27, 2013.

M. Yu and X. Qi. Storage space allocation models for inbound containers in an automatic container terminal. *European Journal of Operational Research*, 226(1):32–45, 2013.